

### **Remarks**

Applicants respectfully request reconsideration of the present application in view of the foregoing amendments and the following remarks. Claims 1 and 3-22 are pending. Claims 1, 11, 15, and 19 are independent. No claims have been allowed. Claims 1-20 are rejected. These rejections are respectfully traversed. Claims 1, 4-6, 10, 11, 15, 19, and 20 have been amended, all for reasons not necessarily related to patentability (e.g., in claim 1, clarifying that the partial procedure summary of the procedure of multithreaded software is embodied on the computer-readable medium). The amendments herein do not necessarily narrow the claims. Claims 21 and 22 have been added. No new matter has been added.

### ***Claim Rejections under 35 U.S.C. § 101***

The Action rejects claims 1 and 3-20 under 35 U.S.C. § 101 as being directed to non-statutory subject matter. These rejections are respectfully traversed.

Claims 1, 11, and 15 have been amended to clarify that the partial procedure summary of the procedure of multithreaded software, the plurality of procedure summaries for the multithreaded software, and the model of the multithreaded software, respectively, are embodied on a computer-readable medium. Therefore, Applicants respectfully submit that independent claims 1, 11, and 15 and all claims depending therefrom produce a “useful, concrete, and tangible result.” Accordingly, Applicants respectfully submit that the 35 U.S.C. § 101 rejections of claims 1 and 3-18 should be withdrawn.

Claims 19 and 20 are directed to a computer-readable medium having encoded thereon a data structure. A memory storing data in a particular data structure is patentable. *See, e.g., In re Lowry*, 32 F.3d 1579 (Fed. Cir. 1994). Also, a “claimed computer-readable medium encoded

with a data structure defines structural and functional interrelationships between the data structure and the computer software and hardware components which permit the data structure's functionality to be realized, and is thus statutory.” (See MPEP § 2106.01(I).) Accordingly, Applicants respectfully submit that the rejections of claims 19 and 20 should be withdrawn.

### ***Claim Rejections under § 112***

The Action rejects claims 1-20 under 35 U.S.C. § 112, second paragraph, as being indefinite. These rejections are respectfully traversed.

Claim 1 has been amended to recite: “wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure.”

For example, the Specification states at page 13, lines 7-18:

FIG. 12 shows an exemplary format 1200 for a procedure summary 1220. In the example, one or more state pairs 1230A-1230N are stored for the summary 1220. For a respective state pair (e.g., consider the state pair 1230A), an initial state 1231A and a resulting state 1235A can be stored. In addition, a starting program counter location 1232A can be stored for the initial state 1231A, and a resulting program counter location 1236A can be stored for the resulting state 1235A.

The state pair 1230A can be used to model execution for the procedure. If the modeled state of the software is the initial state 1231A at location 1232A within the procedure, then the state (or one of the possible states) of the software after execution of the modeled procedure or portion thereof will be the resulting state 1235A at location 1236A. Execution may continue within the procedure and be modeled by another state pair within the procedure summary 1220.

Claim 4 has been amended to recite: “responsive to determining the modeled state fails the indicated state invariant, wherein determining the modeled state fails the indicated state invariant comprises determining that a condition is false for at least one execution path, indicating that a programming flaw is present in the software.” For example, the Specification states at page 7, line 13-17:

For example, one such mechanism is a specified invariant. An example implementation of a specified invariant is called an “assert.” The assert indicates a condition (i.e., and assertion) that is believed to be invariant at a particular location within the software. If the condition is false for any possible set of execution paths, a programming flaw is indicated. In such a case, the assert is said to have failed.

Claim 5 has been amended to recite: “the at least one state pair comprises an initial state and a resulting state, wherein the resulting state comprises at least one of a plurality of possible states of the multithreaded software after execution of the modeled procedure,” “storing an initial program counter location within the modeled procedure for the initial state,” and “storing a resulting program counter location within the modeled procedure for the resulting state.” For example, the Specification states at page 13, lines 7-18:

FIG. 12 shows an exemplary format 1200 for a procedure summary 1220. In the example, one or more state pairs 1230A-1230N are stored for the summary 1220. For a respective state pair (e.g., consider the state pair 1230A), an initial state 1231A and a resulting state 1235A can be stored. In addition, a starting program counter location 1232A can be stored for the initial state 1231A, and a resulting program counter location 1236A can be stored for the resulting state 1235A.

The state pair 1230A can be used to model execution for the procedure. If the modeled state of the software is the initial state 1231A at location 1232A within the procedure, then the state (or one of the possible states) of the software after execution of the modeled procedure or portion thereof will be the resulting state 1235A at location 1236A. Execution may continue within the procedure and be modeled by another state pair within the procedure summary 1220.

Claim 6 has been amended to recite: “the consulting comprises determining possible execution paths within the procedure and using the procedure summary to explore possible states.” For example, the Specification states at page 9, lines 2-4, that “[f]or example, the reachability analysis can determine possible execution paths within the procedure and use the procedure summary to explore possible states.”

Claim 10 has been amended to recite: “the subset comprising less than all of the plurality of actions of the procedure.”

Claim 11 has been amended to recite: “the analyzing comprises checking the modeled multithreaded software for programming flaws,” and “the procedure summaries comprise a plurality of modeled states of the multithreaded software for multithreaded execution of the multithreaded software.” For example, the Specification states at page 5, lines 14-15, that “[f]or example, the model checker 120 can model execution of multithreaded software 112 to detect programming flaws.” The Specification also states, at page 13, lines 7-18:

FIG. 12 shows an exemplary format 1200 for a procedure summary 1220. In the example, one or more state pairs 1230A-1230N are stored for the summary 1220. For a respective state pair (e.g., consider the state pair 1230A), an initial state 1231A and a resulting state 1235A can be stored. In addition, a starting program counter location 1232A can be stored for the initial state 1231A, and a resulting program counter location 1236A can be stored for the resulting state 1235A.

The state pair 1230A can be used to model execution for the procedure. If the modeled state of the software is the initial state 1231A at location 1232A within the procedure, then the state (or one of the possible states) of the software after execution of the modeled procedure or portion thereof will be the resulting state 1235A at location 1236A. Execution may continue within the procedure and be modeled by another state pair within the procedure summary 1220.

Claim 15 has been amended to recite: “a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws.” For example, the Specification states at page 5, lines 14-15, that “[f]or example, the model checker 120 can model execution of multithreaded software 112 to detect programming flaws.”

Claims 19 and 20 have been amended to recite a “computer-readable medium.”

Accordingly, Applicants respectfully request that the rejections of claims 1 and 3-20 under 35 U.S.C. § 112 be withdrawn.

***Cited Art***

Tyrrell *et al.*, “CSP Methods for Identifying Atomic Actions in the Design of Fault Tolerant Concurrent Systems” (hereinafter Tyrrell).

***Patentability of Claims 1 and 3-22 over Tyrrell under 35 U.S.C. § 102(b)***

The Action rejects claims 1 and 3-20 under 35 U.S.C. § 102(b) as being anticipated by Tyrrell. These rejections are respectfully traversed. Applicants respectfully submit that the claims in their present form are allowable over the cited art. For a 102(b) rejection to be proper, the cited art must show each and every element as set forth in a claim. (See MPEP § 2131.01.) However, the cited art does not so show. For example, with respect to claim 1, Tyrrell does not show generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to multithreaded execution of the multithreaded software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure.

***Claims 1, 3-10, 21, and 22***

Claim 1 is directed to a computer program product embodied on a first computer-readable medium and comprising code that, when executed, causes a computer to perform a method of generating a partial procedure summary of a procedure of multithreaded software, wherein the procedure performs a plurality of actions when executed, and recites:

identifying a plurality of the actions as atomically modelable with respect to multithreaded execution of the procedure; and

generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to multithreaded execution of the multithreaded software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure, and

wherein the partial procedure summary of the procedure of multithreaded software is embodied on a second computer-readable medium (emphasis added).

For example, FIG. 12 of the present application shows an exemplary format for a procedure summary, as described at page 13, lines 7-24. In the described example, “[t]he procedure summary 1220 can include a plurality of partial procedure summaries summarizing different sets of actions for the procedure,” where “[a] partial procedure summary for the same set of actions can include a plurality of state pairs to represent a plurality of initial states, a plurality of resulting states, or some combination thereof.”

*Tyrrell does not show generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to multithreaded execution of the multithreaded software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure.* In its rejection of claim 1, the Action relies on various passages in Tyrrell; however, these passages are understood to describe an expansion resulting from a trace evaluation, not “generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to multithreaded execution of the multithreaded software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure” as recited in claim 1.

For example, the Action relies upon Tyrrell at page 630, including Figs. 1 and 2. During its discussion of boundaries within a state space of a set of processes, Tyrrell describes a “fault tolerant mechanism” (see Fig. 1) and action/process-recovery diagrams (see Fig. 2), as well as a boundary “across which error propagation by communication is prevented.” However, one of ordinary skill in the art could not be expected to surmise “generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to

multithreaded execution of the multithreaded software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure” from the mere discussion of such state space boundaries.

Applicants fail to see how a discussion of such state space boundaries would lead one of ordinary skill in the art to the claimed arrangement, which involves “generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to multithreaded execution of the multithreaded software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure,” as recited in claim 1.

The Action also relies, for example, upon Tyrrell at page 633, including Figs. 3 and 4. During its discussion of a trace evaluation, Tyrrell describes a “[s]imple example with three parallel processes” (*see* Fig. 3) and a “[f]inal set of traces” (*see* Fig. 4). However, one of ordinary skill in the art could not be expected to surmise state pairs that model execution for a procedure, let alone the claimed arrangement of “generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to multithreaded execution of the multithreaded software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure” from the mere discussion of a trace evaluation, particularly a trace evaluation for a “[s]imple example with three parallel processes.”

Applicants fail to see how such a trace evaluation for a “[s]imple example with three parallel processes” would lead one of ordinary skill in the art to the claimed arrangement, which involves “generating the partial procedure summary of the procedure from the plurality of the actions atomically modelable with respect to multithreaded execution of the multithreaded

software, wherein the partial procedure summary comprises at least one state pair, wherein the at least one state pair models execution for the procedure,” as recited in claim 1.

Since the cited reference does not show all of the elements recited in claim 1, Applicants believe the claim is not subject to a 102(b) rejection and request the rejection be withdrawn.

Thus, Applicants respectfully submit that claim 1 and its dependent claims 3-10, 21, and 22 are allowable over the cited art.

#### *Claims 11-14*

Claim 11 is directed to a computer program product embodied on a first computer-readable medium and comprising code that, when executed, causes a computer to perform a method of modeling multithreaded software, and recites:

analyzing actions of the multithreaded software, wherein the analyzing comprises checking the modeled multithreaded software for programming flaws; and  
based on the analyzing, generating a plurality of procedure summaries for the multithreaded software, wherein the plurality of procedure summaries for the multithreaded software is embodied on a second computer-readable medium;  
wherein the procedure summaries comprises a plurality of modeled states of the multithreaded software for multithreaded execution of the multithreaded software (emphasis added).

*Tyrrell does not show analyzing actions of the multithreaded software, wherein the analyzing comprises checking the modeled multithreaded software for programming flaws.* In its rejection of claim 11, the Action relies on various passages in Tyrrell; however, these passages are understood to describe an expansion resulting from a trace evaluation, not “analyzing actions of the multithreaded software, wherein the analyzing comprises checking the modeled multithreaded software for programming flaws,” let alone “based on the analyzing, generating a plurality of procedure summaries for the multithreaded software,” as recited in claim 11.



For example, the Action relies upon Tyrrell at page 630, including Figs. 1 and 2. During its discussion of boundaries within a state space of a set of processes, Tyrrell describes a “fault tolerant mechanism” (*see* Fig. 1) and action/process-recovery diagrams (*see* Fig. 2), as well as a boundary “across which error propagation by communication is prevented.” However, one of ordinary skill in the art could not be expected to surmise the claimed arrangement of “analyzing actions of the multithreaded software, wherein the analyzing comprises checking the modeled multithreaded software for programming flaws,” let alone “based on the analyzing, generating a plurality of procedure summaries for the multithreaded software” from the mere discussion of such state space boundaries.

Applicants fail to see how a discussion of such state space boundaries would lead one of ordinary skill in the art to the claimed arrangement, which involves “analyzing actions of the multithreaded software, wherein the analyzing comprises checking the modeled multithreaded software for programming flaws” and “based on the analyzing, generating a plurality of procedure summaries for the multithreaded software,” as recited in claim 11.

The Action also relies, for example, upon Tyrrell at page 633, including Figs. 3 and 4. During its discussion of a trace evaluation, Tyrrell describes a “[s]imple example with three parallel processes” (*see* Fig. 3) and a “[f]inal set of traces” (*see* Fig. 4). However, one of ordinary skill in the art could not be expected to surmise the claimed arrangement of “analyzing actions of the multithreaded software, wherein the analyzing comprises checking the modeled multithreaded software for programming flaws,” let alone “based on the analyzing, generating a plurality of procedure summaries for the multithreaded software” from the mere discussion of a trace evaluation, particularly a trace evaluation for a “[s]imple example with three parallel processes.”

Applicants fail to see how such a trace evaluation for a “[s]imple example with three parallel processes” would lead one of ordinary skill in the art to the claimed arrangement, which involves “analyzing actions of the multithreaded software, wherein the analyzing comprises checking the modeled multithreaded software for programming flaws” and “based on the analyzing, generating a plurality of procedure summaries for the multithreaded software,” as recited in claim 11.

Since the cited reference does not show all of the elements recited in claim 11, Applicants believe the claim is not subject to a 102(b) rejection and request the rejection be withdrawn.

Thus, Applicants respectfully submit that claim 11 and its dependent claims 12-14 are allowable over the cited art.

#### *Claims 15-18*

Claim 15 is directed to a computer program product embodied on a first computer-readable medium and comprising code that, when executed, causes a computer to implement a system for modeling multithreaded software, and recites:

a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws, the model checker comprising:

the model of the multithreaded software, wherein the model comprises a plurality of procedure summaries modeling states of the multithreaded software during multithreaded execution of the multithreaded software, wherein the model of the multithreaded software is embodied on a second computer-readable medium (emphasis added).

For example, FIG. 1 of the present application is a block diagram representing an exemplary system for analyzing multithreaded software via procedure summaries, as described at page 5, lines 11-19. In the example, a model checker analyzes the multithreaded software. For example, the model checker can model execution of multithreaded software to detect programming flaws.

In practice, the multithreaded software can be first converted into a model, which can represent the multithreaded software in a form appropriate to facilitate modeling.

*Tyrrell does not show a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws.* In its rejection of claim 15, the Action relies on various passages in Tyrrell; however, these passages are understood to describe an expansion resulting from a trace evaluation, not “a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws,” as recited in claim 15.

For example, the Action relies upon Tyrrell at page 630, including Figs. 1 and 2. During its discussion of boundaries within a state space of a set of processes, Tyrrell describes a “fault tolerant mechanism” (*see* Fig. 1) and action/process-recovery diagrams (*see* Fig. 2), as well as a boundary “across which error propagation by communication is prevented.” However, one of ordinary skill in the art could not be expected to surmise the claimed arrangement of “a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws” from the mere discussion of such state space boundaries. In fact, Tyrrell’s state space boundaries teach away from “a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws” because “such a boundary of necessity prohibits the passing of information to any process not involved in the atomic action” (*see* Tyrrell at page 630, left column, last paragraph, emphasis in original).

Applicants fail to see how a discussion of such state space boundaries would lead one of ordinary skill in the art to the claimed arrangement, which involves “a model checker operable to

analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws,” as recited in claim 15.

The Action also relies, for example, upon Tyrrell at page 633, including Figs. 3 and 4. During its discussion of a trace evaluation, Tyrrell describes a “[s]imple example with three parallel processes” (*see* Fig. 3). However, one of ordinary skill in the art could not be expected to surmise the claimed arrangement of “a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws” from the mere discussion of a trace evaluation, particularly a trace evaluation for a “[s]imple example with three parallel processes.”

Applicants fail to see how such a trace evaluation for a “[s]imple example with three parallel processes” would lead one of ordinary skill in the art to the claimed arrangement, which involves “a model checker operable to analyze a model of the multithreaded software via checking the model of the multithreaded software for programming flaws,” as recited in claim 15.

Since the cited reference does not show all of the elements recited in claim 15, Applicants believe the claim is not subject to a 102(b) rejection and request the rejection be withdrawn.

Thus, Applicants respectfully submit that claim 15 and its dependent claims 16-18 are allowable over the cited art.

#### *Claims 19-20*

Claim 19 is directed to one or more computer-readable media having encoded thereon a data structure, and recites:

a plurality of state pairs representing a procedure summary for multithreaded software, wherein at least one of the state pairs comprises an initial state and a resulting

state indicating a state after execution of actions modeled by the procedure summary, wherein the procedure summary models multithreaded execution of the multithreaded software (emphasis added).

*Tyrrell does not show a plurality of state pairs representing a procedure summary for multithreaded software.* In its rejection of claim 19, the Action relies on various passages in Tyrrell; however, these passages are understood to describe an action diagram, not “a plurality of state pairs representing a procedure summary for multithreaded software,” as recited in claim 19.

For example, the Action relies upon Tyrrell at page 630, including Fig. 2. With respect to Fig. 2, Tyrrell describes “(a) an action diagram, (b) a process-recovery diagram, and (c) the action diagram dual of [the process-recovery diagram].” However, one of ordinary skill in the art could not be expected to surmise the claimed arrangement of “a plurality of state pairs representing a procedure summary for multithreaded software” from the mere discussion of “(a) an action diagram, (b) a process-recovery diagram, and (c) the action diagram dual of [the process-recovery diagram].”

Applicants fail to see how such “(a) an action diagram, (b) a process-recovery diagram, and (c) the action diagram dual of [the process-recovery diagram]” would lead one of ordinary skill in the art to the claimed arrangement, which involves “a plurality of state pairs representing a procedure summary for multithreaded software,” as recited in claim 19.

Since the cited reference does not show all of the elements recited in claim 19, Applicants believe the claim is not subject to a 102(b) rejection and request the rejection be withdrawn.

Thus, Applicants respectfully submit that claim 19 and its dependent claim 20 are allowable over the cited art.

***Request for Interview***

If any issues remain, the Examiner is formally requested to contact the undersigned attorney prior to issuance of the next Office Action in order to arrange a telephonic interview. It is believed that a brief discussion of the merits of the present application may expedite prosecution. Applicants submit the foregoing formal Amendment so that the Examiner may fully evaluate Applicants' position, thereby enabling the interview to be more focused.

This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.

***Conclusion***

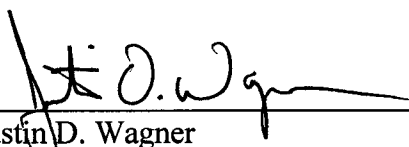
The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

One World Trade Center, Suite 1600  
121 S.W. Salmon Street  
Portland, Oregon 97204  
Telephone: (503) 595-5300  
Facsimile: (503) 595-5301

By

  
Justin D. Wagner  
Registration No. 54,519